Distributed Data Management in Sensor Networks using miniDB / miniSQL

Article

5 authors, including:

Abdalkarim Awad
Birzeit University
28 PUBLICATIONS **239** CITATIONS

SEE PROFILE

Falko Dressler
Universität Paderborn
287 PUBLICATIONS **3,469** CITATIONS

SEE PROFILE

Distributed Data Management in Sensor Networks using miniDB / miniSQL

Abdalkarim Awad, Wei Xie, Eugen Rose, Reinhard German and Falko Dressler
Computer Networks and Communication Systems
Dept. of Computer Science, University of Erlangen, Germany

***Abstract*—Recent research in the domain of sensor networks demonstrated the strong need for network-centric data pre-processing in order to maximize the lifetime of typical sensor networks. This includes the feasibility to store and to retrieve data close to the origin of the measurements. We have build a database oriented approach to sensor networks that is primarily focusing on minimizing the communication overhead. In particular, an API, named miniDB, allows to store and retrieve named data in sensor nodes and an associated communication protocol, named miniSQL, can be used to identify available data and to retrieve them from any point in the network. SQL-like expressions can be used for simplified access to the data, either addressing single nodes or tables, or even the entire network. In conclusion, the system represents all necessary base functions for distributed data management operations in a network-centric manner.**

I. Introduction

Wireless Sensor Networks (WSNs) are representing a wide variety of research challenges. Some properties are well established in the sensor networks community. This includes strong resource limitations in terms of CPU, energy, and bandwidth. Especially the last two constraints are heavily interacting. In general, the communication is one of the most energy consuming operations of sensor nodes. Even though different numbers can be found in the literature, it can be assumed that more that 90 % of the energy load correspond to communication, including retransmissions due to congestion and other side effects. Thus, it is well agreed that minimizing the communication load greatly affects the lifetime of the sensor network. Looking at many WSN application such as surveillance, burglar alarms, agricultural applications and others, the original concept of constant observation of all measures is usually not necessary. Instead, the WSN can be programmed to transmit data only in the case of alarms, i.e. exceeded threshold values.

Nevertheless, the operator of the network often wants to get more detailed information even from time intervals in which the thresholds have not been exceeded. This basically contradicts all approaches to save energy and to maximize the network lifetime. We propose a concept for data management in WSNs that is inspired by database technology. The basic idea is to enable single sensors to store as much data as necessary (or possible) in order to provide a broad view on recent measurements. Nevertheless, this data is never transmitted through the network if not explicitly demanded. The data can be identified by associated names. From an application point of view, SQL-like expressions can be used for simplified access to the data, either addressing single nodes or tables, or even the entire network. In conclusion, the system represents all necessary base functions for distributed data management operations in a network-centric manner.

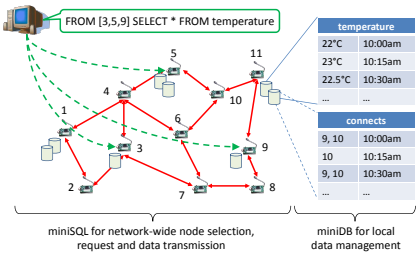

Fig. 1. Scenario for network-centric storage and retrieval of named data items using SQL-like expression

II. An Architecture for Data Management

In order to optimize the data retrieval in WSNs, a number of approaches have been proposed in recent papers. The following architectures can be identified:

Centralized management – All data items are immediately after generation transmitted to a centralized system, i.e. the base station. This is the standard architecture in most WSN scenarios but we will not elaborate on this for given reasons.

Semi-distributed architecture – In this case, the entire network is considered a distributed data base and sensor nodes pre-process data. The best known approaches are TinyDB developed at UC Berkeley [1] and Cougar from Cornell [2], [3]. Both approaches provide a SQL-like interface and optimize the communication overhead by in-network data processing. In particular, aggregation can be performed or thresholds can be checked. Nevertheless, it is not possible to access historical data items as available before aggregating or checking.

Distributed architecture – The key idea is to store data items at (geographically) meaningful places. Available approaches such as Geographic Hash Tables (GHTs) [4], [5] associate events to a hash value. Then, the data items are stored at nodes responsible for the hash value. In order to retrieve the items, the application must be able to perform the same translation.

III. miniDB

The miniDB systems is responsible for storing data in a structured way locally in a sensor node. In particular, we implemented the system for the BTnode architecture developed at

ETH Zurich. Using the available API, data items can be stored, located, retrieved, and removed in a tabular approach. Names are associated as identifiers to the data items in order to ease access. This meta information increases the necessary memory only minimal but provides a great opportunity to locate and to retrieve data from arbitrary nodes, even from dynamically placed new nodes, in a simplified manner. In particular, the the API operations depicted in Figure 2 allow to the following operations: tables can be created and dropped, records can be inserted, identified, and discarded. Additionally, all available meta information can be retrieved.

```
unsigned int create_table(char *table_name, unsigned int
      content_size, unsigned int column_counter, struct
      column_info *column_infos)
unsigned int insert(char *table_name, unsigned int *
      allocationtable_index, ...)
int *select(char *table_name, char *attribute, char *
      operation, char *operand);
int *multi_select(char *table_name, int log, char **logic,
      struct condition *conditions);
int delete_with_array (char * table_name, int *ret_array);
int delete_with_condition(char *table_name, int log, char
      **logic, struct condition *conditions);
```

Fig. 2. Available API calls for miniDB

IV. miniSQL

In order to access the data stored locally at distributed nodes, miniSQL has been developed. In principle, miniSQL is a client-server architecture. In our lab implementation, we currently support communication over heterogeneous communication links, i.e. Bluetooth and low-power radio using BMAC. Multi-hop functionality is provided as well as proxying between the different radio models. The name miniSQL was coined to represent the shape of data queries. A SQL-like syntax allows to query single nodes as well as multiple (up to all) nodes in the network simultaneously. Figure 3 depicts the syntax including two sample queries outlining the principles of the system. The optional `FROM` at the beginning selects the sensor nodes to be queried. The query is completed using a standard `SELECT` operation, i.e. specifying attributes, tables, and clauses.

```
[ FROM nodeID ] SELECT * | attribute [ attribute ] FROM
      tableName [ WHERE clause ]
FROM 1 SELECT * FROM ROSES WHERE light > 200
SELECT * FROM ROSES WHERE temp < 30 and time > 11-00-00
```

Fig. 3. miniSQL syntax and examples

For transmitting the messages among the nodes in the network, a transport protocol has been developed that explicitly supports the SQL-like queries of named data items. According to the query syntax, the protocol fields are mapped to data packets as shown in Figure 4. miniSQL allows the querier to connect to arbitrary sensors in the field and to transmit the optimized query. The receiving node checks whether the request is addressed to itself and whether the queried data is available. Then, it transmits the results back to the querier.

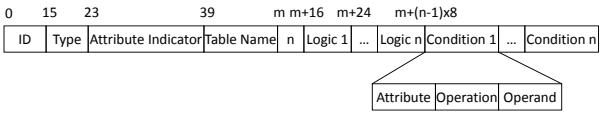

Fig. 4. Frame format for transmission of queries between sensor nodes

The ID of the system has a size of two byte following the BMAC address field. The type field indicates the message being a query or a response. Then the attribute indicator (a bit field of 2 byte) specifies the attributes that should be sent back to the querying system. The table name is a string identifying particular data items. Conditions consist of three fields: an attribute selecting a field in the table, an operator, and a value to compare with. The number of conditions is prepended to these fields.

V. Operation Example

In our lab, we established and tested a smart home environment in order to perform certain tasks, such as monitoring and control of domestic systems [6]. We are integrating the miniDB/miniSQL system to collect data from a central diagnosis center or stations able to analyze and monitor the patient's behavior. For example, unused active systems like lights can be identified, analyzed, and controlled. Additionally, collected data can be correlated. Concerning the sensor network, the operation is subject to certain technical constraints. So the sensor nodes are only conditionally reliable. Furthermore, new functionality for maintenance, management, and diagnosis ought to be integrated.

VI. Conclusion

With the developed system, we fill the gap between semi-distributed systems that perform continuous in-network data pre-processing such as TinyDB and Cougar and the requirement to obtain historic data from selected measurements. This transmission is only performed if required and the network is responsible to locate and to retrieve the data items.

References

[1] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, March 2005.

[2] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *ACM SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.

[3] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao, "The Cougar Project: a work-in-progress report," *ACM SIGMOD Record*, vol. 32, no. 4, pp. 53–59, 2003.

[4] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," in *1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, September 2002.

[5] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-Centric Storage in Sensornets," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 137–142, January 2003.

[6] S. Dengler, A. Awad, and F. Dressler, "Sensor/Actuator Networks in Smart Homes for Supporting Elderly and Handicapped People," in *21st IEEE International Conference on Advanced Information Networking and Applications (AINA-07): First International Workshop on Smart Homes for Tele-Health (SmarTel'07)*, vol. II. Niagara Falls, Canada: IEEE, May 2007, pp. 863–868.